

Course Project

Due: Wednesday, April 12, 4pm

Part 1 In this part, we will use the kinematic car model presented in [1, Section 4.1.1] (references are listed on the last page of this document). You will first create a Simulink diagram, then implement two closed-loop control laws.

- (a) The dynamics of the system are governed by

$$\begin{aligned}\dot{x} &= v \cos \theta \\ \dot{y} &= v \sin \theta \\ \dot{\theta} &= \frac{v}{L} \tan \gamma\end{aligned}\tag{1}$$

Set up a Simulink subsystem which inputs rear wheel velocity v and steering angle γ , and outputs the planar position (x, y) and heading angle θ . Take $L = 1$ m. Provide a screenshot of your Simulink design, including the contents of any subsystems you used.

- (b) Create a closed-loop control system to move to a desired point. This system inputs a desired planar position $(x_{\text{ref}}, y_{\text{ref}})$, computes the reference heading

$$\theta_{\text{ref}} = \text{atan2}(y_{\text{ref}} - y, x_{\text{ref}} - x)$$

then employs the control

$$\begin{aligned}v &= k_p^v \sqrt{(x_{\text{ref}} - x)^2 + (y_{\text{ref}} - y)^2} \\ \gamma &= k_p^\gamma (\theta_{\text{ref}} \ominus \theta)\end{aligned}$$

where $\theta_{\text{ref}} \ominus \theta$ is the shortest path from θ_{ref} to θ on the unit circle — this is discussed in the course notes. Using an initial state of $(x_0, y_0, \theta_0) = (0, 0, \pi/4)$, tune the control gains k_p^v and k_p^γ to drive the car to the reference (desired) position $(x_{\text{ref}}, y_{\text{ref}}) = (10, -8)$ m. Provide a plot of the states (x, y, θ) and inputs (v, γ) as well as the values of the control gains you used.

- (c) Using the same initial and desired state as (b), tune the gains k_p^v and k_p^γ to make the vehicle drive continuously around the goal without reaching it — this is known as a limit cycle. Provide the values of the gains used along with a plot of x versus y (i.e. overhead position of the car) for your limit cycle.
- (d) We will now implement a controller proposed by [2] allowing the kinematic car to move to a desired 2D pose $(x_{\text{ref}}, y_{\text{ref}}, \theta_{\text{ref}})$. Introduce the new input ω through

$$\gamma = \text{atan2}(\omega L, v)\tag{2}$$

which transforms (1) into

$$\begin{aligned}\dot{x} &= v \cos \theta \\ \dot{y} &= v \sin \theta \\ \dot{\theta} &= \omega\end{aligned}\tag{3}$$

Define the set of variables

$$\begin{aligned}\rho &= \sqrt{(x_{\text{ref}} - x)^2 + (y_{\text{ref}} - y)^2} \\ \phi &= \text{atan2}(y_{\text{ref}} - y, x_{\text{ref}} - x) \\ \alpha &= \phi - \theta \\ \beta &= \phi - \theta_{\text{ref}}\end{aligned}$$

Using the nonlinear feedback control law

$$\begin{aligned}v &= k_v \rho \cos \alpha \\ \omega &= k_\omega \alpha + k_v \frac{\sin \alpha \cos \alpha}{\alpha} (\alpha + \beta)\end{aligned}$$

with $k_v > 0$, $k_\omega > 0$ can be shown (see course notes) to make (3) converge asymptotically to $(x_{\text{ref}}, y_{\text{ref}}, \theta_{\text{ref}})$. Note because our actual system (1) employs input γ instead of ω , we employ the above result in (2) to obtain γ . Implement the above-described design in Simulink (you may wish to use the **MATLAB Function** block found in Library Browser under **Simulink -> User-Defined Functions** for the control law). Provide a screenshot of your design, plus a print-out of your MATLAB function if you used it.

- (e) Using the design in (d) and initial pose $(x_0, y_0, \theta_0) = (0, 0, 0)$, use your design to drive to the following final configurations, tuning gains k_v , k_ω to get smooth performance:

- (i) $(x_{\text{ref}}, y_{\text{ref}}, \theta_{\text{ref}}) = (5, 8, 0)$
- (ii) $(x_{\text{ref}}, y_{\text{ref}}, \theta_{\text{ref}}) = (-10, -4, -\pi/2)$

For each run, provide plots of the vehicle states (x, y, θ) and inputs (v, γ) plus the values of control gains you used for each run.

- (f) During your gain tuning for this second design, did the system perform a limit cycle?

Part 2 Load the satellite Simulink diagram provided to you on eClass. This system inputs torques τ_1 , τ_2 , τ_3 about its three body-fixed axes, and outputs attitude $R \in SO(3)$ and body-frame angular velocity $\omega_B \in \mathbb{R}^3$. Note the output attitude is a matrix signal which gets converted to Euler angles (ϕ, θ, ψ) using a custom **MATLAB Function** block named **R2e**.

- (a) First run the system in open-loop mode, inputting a step input of magnitude 0.1 N m along an individual axis, then scoping the output angular velocity vector ω_B . You should observe a “cross-coupling” effect between the output axes. What is the cause of this? How could the satellite be structurally re-designed to avoid this effect?
- (b) Now implement the linear proportional-derivative control law

$$\begin{aligned}\tau_1 &= k_p^1(\phi_{\text{ref}} - \phi) + k_d^1(-\dot{\phi}) \\ \tau_2 &= k_p^2(\theta_{\text{ref}} - \theta) + k_d^2(-\dot{\theta}) \\ \tau_3 &= k_p^3(\psi_{\text{ref}} - \psi) + k_d^3(-\dot{\psi})\end{aligned}$$

where the Euler angle time derivatives $(\dot{\phi}, \dot{\theta}, \dot{\psi})$ can be obtained from

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi \sec \theta & \cos \phi \sec \theta \end{bmatrix} \begin{bmatrix} \omega_{B,1} \\ \omega_{B,2} \\ \omega_{B,3} \end{bmatrix}$$

Provide a screenshot of your design. What is the reason for including a derivative feedback action?

- (c) Starting from initial conditions $R(0) = I \implies (\phi(0), \theta(0), \psi(0)) = (0, 0, 0)$, tune the linear control gains in order for the system to stabilize an attitude of $(\phi_{\text{ref}}, \theta_{\text{ref}}, \psi_{\text{ref}}) = (\pi/6, \pi/6, \pi/6)$. Provide plots of (i) Euler angles versus time and (ii) torque inputs versus time during this maneuver.
- (d) Using the same control gains as (c), try stabilizing $(\phi_{\text{ref}}, \theta_{\text{ref}}, \psi_{\text{ref}}) = (2\pi/3, 2\pi/3, 2\pi/3)$. Provide the same plots as in (c) and explain what you're seeing. Can performance be improved by re-tuning the gains, and why?
- (e) Now replace the linear closed-loop control with the nonlinear controller described in [3],

$$\tau = -K_v \omega_B - K_p \sum_{i=1}^3 \alpha_i e_i \times (R_{\text{ref}}^T R e_i)$$

where $K_v, K_p \in \mathbb{R}^{3 \times 3}$ are positive-definite matrices (for instance diagonal matrices with positive, non-zero entries), $\alpha_k \in \mathbb{R}^+$ are positive, non-zero and *distinct* scalars, $e_1 = [1 \ 0 \ 0]^T$ etc, and $R_{\text{ref}}, R \in SO(3)$ are the desired and actual attitude, respectively. Suggestion: use the **MATLAB Function** block to implement your control law. Provide a screenshot of your design and a print-out of your MATLAB code if you used it.

- (f) Tune the nonlinear controller to stabilize R_{ref} corresponding to $(\phi_{\text{ref}}, \theta_{\text{ref}}, \psi_{\text{ref}}) = (\pi/6, \pi/6, \pi/6)$. Provide plots of Euler angles versus time during this maneuver. How does performance compare with the linear control from (c)?
- (g) Now run the nonlinear controller using R_{ref} corresponding to $(\phi_{\text{ref}}, \theta_{\text{ref}}, \psi_{\text{ref}}) = (2\pi/3, 2\pi/3, 2\pi/3)$. Provide plots of Euler angles versus time, and compare against (d). Can performance be improved by re-tuning the gains of the nonlinear controller?

Part 3 Load the quadrotor Simulink diagram provided on eClass. This system is a 6 DoF rigid-body with inputs T (net thrust) and $\tau_\phi, \tau_\theta, \tau_\psi$ (torques along roll, pitch and yaw axes, respectively), and outputs $R \in SO(3)$ (attitude), ω (angular velocity vector in the body-fixed frame), and p, v (position and velocity with respect to the ground-fixed North-East-Down frame). Remark $T < 0$ corresponds to an upward thrust due to b_3 pointing down, and that R is converted to roll-pitch-yaw Euler angles (ϕ, θ, ψ) . You will need to manually rotate the Simscape visualization perspective to orient the third reference axis into the down position.

- (a) Implement the linear closed-loop control described in [1, Section 4.2]. This control consists of three proportional-derivative controllers working in tandem: The **lateral and longitudinal** control

$$\begin{aligned}\tau_r &= k_p^r(\phi_{\text{ref}} - \phi) - k_d^r \dot{\phi} \\ \tau_p &= k_p^p(\theta_{\text{ref}} - \theta) - k_d^p \dot{\theta}\end{aligned}$$

where $\dot{\phi}, \dot{\theta}$ are obtained from ω_B as in the satellite controller, and $(\phi_{\text{ref}}, \theta_{\text{ref}})$ are computed by

$$\begin{aligned}\phi_{\text{ref}} &= k_p^\phi(p_{\text{ref}}^{\text{lat}} - p^{\text{lat}}) - k_d^\phi v_{\text{lat}} \\ \theta_{\text{ref}} &= -k_p^\theta(p_{\text{ref}}^{\text{lon}} - p^{\text{lon}}) + k_d^\theta v_{\text{lon}}\end{aligned}$$

where

$$\begin{bmatrix} p^{\text{lat}} \\ p^{\text{lon}} \end{bmatrix} = \underbrace{\begin{bmatrix} -\sin \psi & \cos \psi \\ \cos \psi & \sin \psi \end{bmatrix}}_{R_\psi} \begin{bmatrix} p^N \\ p^E \end{bmatrix}$$

and

$$\begin{bmatrix} p_{\text{ref}}^{\text{lat}} \\ p_{\text{ref}}^{\text{lon}} \end{bmatrix} = R_\psi \begin{bmatrix} p_{\text{ref}}^N \\ p_{\text{ref}}^E \end{bmatrix}, \quad \begin{bmatrix} v^{\text{lat}} \\ v^{\text{lon}} \end{bmatrix} = R_\psi \begin{bmatrix} v^N \\ v^E \end{bmatrix}$$

with $[p^N \ p^E]^T$, $[v^N \ v^E]^T$ the vehicle position and velocity in the NED frame, and $[p_{\text{ref}}^N \ p_{\text{ref}}^E]^T$ the reference trajectory in the NED frame; the **yaw** control

$$\tau_y = k_p^y(\psi_{\text{ref}} \ominus \psi) - k_d^y \dot{\psi}$$

where ψ_{ref} is the reference yaw angle of the vehicle and \ominus is the same operator as in Part 1b; and the **altitude** control

$$T = k_p^a(p_{\text{ref}}^D - p^D) - k_d^a v^D - mg$$

where $T < 0$ corresponds to an upwards thrust, p^D and v^D are the vehicle's position and velocity along the down axis, $m = 1.216$ kg is the mass of the vehicle, and $g = 9.81$ m/s². Provide a screenshot of your design.

- (b) Using your control from (a) and zero initial conditions (representing a take-off configuration with the quadrotor pointing north), tune your gains to stabilize $[p_{\text{ref}}^N \ p_{\text{ref}}^E \ p_{\text{ref}}^D \ \psi_{\text{ref}}] = [1 \ 1 \ -2 \ \pi/2]$ i.e. a hover 2 m above the ground facing E. Your gains should be tuned to avoid any significant overshoots. Provide plots of position (as p^N , p^E and p^D) and attitude (as ϕ , θ and ψ) versus time during this maneuver.
- (c) Starting from an initial hover at $[p_0^N \ p_0^E \ p_0^D] = [0 \ 0 \ -2]$ with attitude $[\phi_0 \ \theta_0 \ \psi_0] = [0 \ 0 \ 0]$, tune the gains to track a figure-8 trajectory in space described by the parametric curves

$$\begin{aligned} p_{\text{ref}}^N(t) &= l_M \sin(2\pi t/t_c) \\ p_{\text{ref}}^E(t) &= (l_m/2) \sin(4\pi t/t_c) \\ p_{\text{ref}}^D(t) &= (-h/2) \sin(\pi t/t_c) - 2 \end{aligned}$$

where $l_M = 2$, $l_m = 1$ are the major and minor diameters of each lobe, $h = 2$ is the total vertical height of the trajectory, and $t_c = 10$ s is the time required to complete one full circuit. Use the reference yaw angle

$$\psi_{\text{ref}}(t) = \text{atan2} \left[(d/dt)p_{\text{ref}}^E(t), (d/dt)p_{\text{ref}}^N(t) \right]$$

which points the vehicle in the direction of travel, and where the time derivatives can be obtained by analytically differentiating the parametric curves. The initial condition on p^D can be specified by going inside the quadrotor subsystem, double-clicking the 6-DOF Joint, expanding Z Prismatic Primitive (Pz) \rightarrow State Targets \rightarrow Specify Position Target (this should be enabled), and then entering the desired p_0^D into the Value field.

Provide a screenshot of your design. Plot positions p and reference positions p_{ref} versus time, attitude (ϕ, θ, ψ) and reference attitude $(\phi_{\text{ref}}, \theta_{\text{ref}}, \psi_{\text{ref}})$ versus time, and control inputs $(T, \tau_r, \tau_p, \tau_y)$ versus time, for $0 \leq t \leq t_c$ i.e. one complete circuit. Note you can **unwrap** the ψ data prior to plotting in order to remove jumps of 2π . The controller gains should be tuned to provide accurate tracking and avoid overshoots.

- (d) Now replace the linear control from (a) by the geometric tracking control law proposed in [4]. First, calculate the vector

$$f = k_p(p_{\text{ref}} - p) + k_v(\dot{p}_{\text{ref}} - v) + m\ddot{p}_{\text{ref}} - mge_3$$

where $k_p, k_v \in \mathbb{R}^+$ are positive constants, $p_{\text{ref}} = [p_{\text{ref}}^N \ p_{\text{ref}}^E \ p_{\text{ref}}^D]^T$ and $p = [p^N \ p^E \ p^D]^T$ are the desired and actual position vector in the world frame, $\dot{p}_{\text{ref}} = (d/dt)p_{\text{ref}}$ and v are the desired and actual velocity vector in the world frame, $\ddot{p}_{\text{ref}} = (d^2/dt^2)p_{\text{ref}}$ is the desired acceleration vector in the world frame, and m is the mass of the vehicle and g is the acceleration due to gravity as in (a). Note that \dot{p}_{ref} and \ddot{p}_{ref} are obtained by analytic differentiation of the parametric curve p_{ref} given in (c). The net thrust input T is calculated as

$$T = f \cdot Re_3$$

where $R \in SO(3)$ is the current attitude of the vehicle and $e_3 = [0 \ 0 \ 1]^T$. As before, $T < 0$ indicates an upward thrust, since $Re_3 = b_3$ points down. The remaining three inputs $\tau_\phi, \tau_\theta, \tau_\psi$ are employed to point the body-fixed frame vector b_3 in the direction $-f$ as follows. Define the reference (desired) attitude matrix

$$SO(3) \ni R_{\text{ref}} = [v_1 \ v_2 \ v_3]$$

The right-most column v_3 , which represents the third body-fixed axis of the quadrotor at its reference attitude expressed in ground-frame coordinates, is assigned as

$$v_3 = -\frac{f}{\|f\|}$$

Columns v_1 and v_2 , representing the ground-frame coordinates of the first and second body-fixed axes of the quadrotor at its reference attitude, lie in the plane normal to v_3 . We will employ the reference yaw angle ψ_{ref} to find their direction using the following series of calculations. First, calculate the two reference vectors

$$\begin{aligned} v_{\text{ref}}^A &= R_3(\psi_{\text{ref}})e_1 = \begin{bmatrix} \cos \psi_{\text{ref}} & -\sin \psi_{\text{ref}} & 0 \\ \sin \psi_{\text{ref}} & \cos \psi_{\text{ref}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos \psi_{\text{ref}} \\ \sin \psi_{\text{ref}} \\ 0 \end{bmatrix} \\ v_{\text{ref}}^B &= R_3(\psi_{\text{ref}})e_2 = \begin{bmatrix} \cos \psi_{\text{ref}} & -\sin \psi_{\text{ref}} & 0 \\ \sin \psi_{\text{ref}} & \cos \psi_{\text{ref}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -\sin \psi_{\text{ref}} \\ \cos \psi_{\text{ref}} \\ 0 \end{bmatrix} \end{aligned}$$

Note that v_2 is perpendicular to both v_3 and v_{ref}^A , while v_1 is perpendicular to both v_3 and v_{ref}^B . The first relationship uniquely determines v_2 except when $v_3 \parallel v_{\text{ref}}^A$ (singularity type “A”), and the second relationship uniquely determines v_1 except when $v_3 \parallel v_{\text{ref}}^B$

(singularity type “B”); however, these two singularities will never occur at the same time. We thus compute the vector norms $\|v_3 \times v_{\text{ref}}^A\|$ and $\|v_{\text{ref}}^B \times v_3\|$, then perform calculations corresponding to the larger of the two values, namely

$$\text{if } \|v_3 \times v_{\text{ref}}^A\| > \|v_{\text{ref}}^B \times v_3\| \implies v_2 = \frac{v_3 \times v_{\text{ref}}^A}{\|v_3 \times v_{\text{ref}}^A\|}, \quad v_1 = v_2 \times v_3$$

or

$$\text{if } \|v_3 \times v_{\text{ref}}^B\| > \|v_{\text{ref}}^A \times v_3\| \implies v_1 = \frac{v_{\text{ref}}^B \times v_3}{\|v_{\text{ref}}^B \times v_3\|}, \quad v_2 = v_3 \times v_1$$

The net result is the reference attitude matrix $R_{\text{ref}} = [v_1 \ v_2 \ v_3]$. This is tracked by the actual attitude $R \in SO(3)$ as follows: define the attitude and angular velocity errors

$$e_R = \frac{1}{2} S^{-1} (R^T R_{\text{ref}} - R_{\text{ref}}^T R)$$

$$e_\omega = R^T R_{\text{ref}} \omega_{\text{ref}} - \omega$$

where $\omega \in \mathbb{R}^3$ is the angular velocity vector of the quadrotor expressed in the body-fixed frame, and ω_{ref} will be discussed shortly. The vector of torque inputs is calculated as

$$\begin{bmatrix} \tau_r \\ \tau_p \\ \tau_y \end{bmatrix} = k_R e_R + k_\omega e_\omega + \omega \times J \omega + J [R^T R_{\text{ref}} \dot{\omega}_{\text{ref}} - S(\omega) R^T R_{\text{ref}} \omega_{\text{ref}}]$$

where $k_R, k_\omega \in \mathbb{R}^+$ are positive constants, J is the mass moment of inertia matrix of the quadrotor, in our case

$$J = \begin{bmatrix} 0.0202 & 0 & 0.004 \\ 0 & 0.0207 & 0 \\ 0.004 & 0 & 0.0356 \end{bmatrix} \text{ kg m}^2$$

and $\dot{\omega}_{\text{ref}} \in \mathbb{R}^3$ is discussed below.

The vector ω_{ref} represents the rate of change of the reference attitude matrix R_{ref} and is obtained from

$$\dot{R}_{\text{ref}} = R_{\text{ref}} S(\omega_{\text{ref}}) \implies \omega_{\text{ref}} = S^{-1} (R_{\text{ref}}^T \dot{R}_{\text{ref}})$$

The vector $\dot{\omega}_{\text{ref}} = (d/dt)\omega_{\text{ref}}$ can be obtained by differentiating the reference attitude matrix dynamics $\dot{R}_{\text{ref}} = R_{\text{ref}} S(\omega_{\text{ref}})$:

$$\ddot{R}_{\text{ref}} = \dot{R}_{\text{ref}} S(\omega_{\text{ref}}) + R_{\text{ref}} S(\dot{\omega}_{\text{ref}}) \implies \dot{\omega}_{\text{ref}} = S^{-1} (R_{\text{ref}}^T \ddot{R}_{\text{ref}} - R_{\text{ref}}^T \dot{R}_{\text{ref}} S(\omega_{\text{ref}}))$$

As shown in the course notes, the expression for ω_{ref} is very complicated, and the situation is much worse for $\dot{\omega}_{\text{ref}}$. Because of this, it's suggested to take

$$\omega_{\text{ref}} = \dot{\omega}_{\text{ref}} = [0 \ 0 \ 0]^T$$

Implement the above-discussed nonlinear control in Simulink. Provide a print-out of your controller code within the **MATLAB Function** block.

- (e) Test your nonlinear control design with the Figure-8 reference trajectory from (c), using the initial conditions $[p_0^N \ p_0^E \ p_0^D] = [0 \ 0 \ -4]$ and $[\phi_0 \ \theta_0 \ \psi_0] = [0 \ \pi \ 0]$ — note this represents a quadrotor initially upside-down. The initial height is specified as in section (c). To specify the initial attitude, go into the quadrotor subsystem, double-click the **6-DOF Joint**, expand **Spherical Primitive (S)**, then set the values as follows:

Field	Value
Specify Position Target	[Enabled]
Priority	High (desired)
Value	Rotation Sequence
Rotation About	Base Axes
Sequence	X-Y-Z
Angles	[0 pi 0] rad

Provide plots of positions p and p_{ref} versus time, attitudes (ϕ, θ, ψ) versus time, and control inputs $(T, \tau_r, \tau_p, \tau_y)$ versus time for one complete circuit $0 \leq t \leq t_c$, as well as the values of controller gains k_p, k_v, k_R, k_ω you used to obtain a stable closed-loop system response. Could the linear control law from (a) be used to perform this experiment? Why or why not?

References

- [1] Peter Corke. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*, volume 118 of *Springer Tracts in Advanced Robotics*. Springer, second edition, 2017.
- [2] Michele Aicardi, Giuseppe Casalino, Antonio Bicchi, and Aldo Balestrino. Closed loop steering of unicycle-like vehicles via Lyapunov techniques. *IEEE Robotics & Automation Magazine*, 2(1):27–35, March 1995.
- [3] Nalin A. Chaturvedi, Amit K. Sanyal, and N. Harris McClamroch. Rigid-body attitude control: Using rotation matrices for continuous, singularity-free control laws. *IEEE Control Systems Magazine*, 31(3):30–51, June 2011.
- [4] Taeyoung Lee, Melvin Leok, and N. Harris McClamroch. Geometric tracking control of a quadrotor UAV on SE(3). In *Proceedings of the 49th IEEE Conference on Decision and Control*, pages 5420–5425, Atlanta, GA, December 2010.